Procedia Computer Science

Volume 29, 2014, Pages 2391–2400

ICCS 2014. 14th International Conference on Computational Science

Pseudorandom Number Generation in the Context of a 3D Simulation Model for Tissue Growth

Belgacem Ben Youssef[1]*and Rachid Sammouda[2]

[1] King Saud University, College of Computer & Information Sciences
Department of Computer Engineering
Riyadh, Saudi Arabia
BBenyoussef@ksu.edu.sa

[2] King Saud University, College of Computer & Information Sciences
Department of Computer Science
Riyadh, Saudi Arabia
RSammouda@ksu.edu.sa

Abstract

In this paper, we consider our choice of a pseudorandom number generator (PRNG) in the context of running a simulation model for the growth of 3D tissues. This PRNG is the multiplicative linear congruential generator (MLCG) with carefully chosen parameters. We base our selection of this generator on three criteria. They are periodicity, randomness quality, and ease of implementation. In these regards, we review some of the pertinent theoretical properties of the employed MLCG and describe techniques used to obtain such sequences serially. Our investigation indicates that the MLCG, with properly selected parameters, can be a good, portable, user-specified, and user-controlled generator with acceptable quality. During the simulation of tissue growth, our various experiments have also shown that the ratio of the total number of random numbers consumed till confluence to the total number of computational sites in the cellular array never exceeds a certain number. This number can be used as a predictor for the period of a PRNG needed to run a particular experiment to simulate tissue growth and to estimate when a longer period may be required in order to deal with very large data sets.

Keywords: Pseudorandom number generation, MLCG, 3D computational model, tissue growth

1 Introduction

The growth of three-dimensional tissues with proper structure and function is the main goal of tissue engineering. Tissue engineers draw on the knowledge gained in the fields of biology, biochemistry, engineering, and the medical sciences to develop bioartifcial implants or to induce tissue remodeling in order to replace, repair or enhance the function of a particular tissue or organ [1, 2]. The development of computational and simulation models for studying biocomplexity at the cell population and tissue level

*Corresponding author.

Selection and peer-review under responsibility of the Scientific Programme Committee of ICCS 2014

doi:10.1016/j.procs.2014.05.223

can provide powerful frameworks in this area, particularly by employing systems-based approaches [3]. These approaches consider cells as system components that migrate, proliferate and interact to generate the complex behavior observed in living systems [4]. However, employing systems-based approaches could lead to models with high complexity whose solution poses significant computational challenges [5]. The availability of computational models with predictive abilities could greatly speed up progress in this area by assisting scientists in predicting the dynamic response of cell populations to external stimuli, and by rapidly assessing the effect of various system parameters on the overall tissue growth rates. Computer simulations can thus be used to shorten the development stage by allowing researchers to quickly screen many alternatives and choose only the most promising ones for laboratory experimentation.

Various modeling approaches have been used to simulate the population dynamics of proliferating cells. Early attempts to model cell population growth were limited to nonmotile cells and to the study of contact inhibition phenomena on the proliferation of anchorage-dependant endothelial cells. These early models considered nonmotile cells proliferating in two dimensions [6], or on microcarriers [7]. In ignoring cell locomotion, the cell growth rates in these early models were restricted by the effects of contact inhibition. Lee et al. in [8] showed the importance of cell motility and cell-cell interaction in describing the cell proliferation rates. Any comprehensive model for tissue growth must consider these processes and account for the growth factors that regulate their rates. Later, Ben Youssef et al. developed a three-dimensional cellular automata model for tissue growth [9]. This work utilizes a Markov chain approach to model the trajectories of migrating cells and is focused primarily on the study of a single population of proliferating and migrating cells [10]. It forms the foundation for our extended model that includes multiple cell types [11].

Pseudorandom numbers (PRN) are useful in many applications including different types of simulations, sampling, numerical analysis, computer programming, decision making, and recreation. Any algorithm that employs random numbers is given the name *Monte Carlo* method, in honor of the European resort town to which random processes are so important [12]. We rely on them extensively in our computational model developed to simulate the growth of three-dimensional tissues. In our algorithm, initially cells are randomly seeded; then are also randomly selected to perform predefined procedures and to reset the state of a site occupied by a living cell. Our sequential algorithms use often a pseudorandom number generator (PRNG) to effect these computations and allow the simulation of tissue growth to proceed according to the cellular automaton rules for cell division, migration, and collision. Our objective in this paper is to report on our experiences using a particular PRNG in a simulation model of tissue growth and to share our reasons and arguments for doing so. We begin by briefly reviewing our stochastic model for tissue growth. We then present some related theoretical background on the PRNG followed by discussions on its periodicity, its randomness quality, and its serial implementation. Section 7 addresses the consumption of random numbers within the simulation model and the possibility of exhausting its period. Finally, section 8 concludes the paper and offers a summary of future work.

2 Stochastic Model for Tissue Growth

As an outcome of the biological processes of cell division, cell motion, and cell collision, we model the process of tissue growth by considering the following stages:

1. Proliferating cells execute persistent random walks in space [13]. Each cell in the population moves in one direction for a certain period of time (persistence). At the end of this interval, the cell stops and turns to continue its migration in another direction. The persistence is a random variable whose density function can be determined experimentally.

2. When two cells collide, they stop for a short period of time before resuming their migration to move away from each other.

3. At the end of its cycle, a cell stops to divide into two daughter cells. The cell cycle or division time is another random variable whose density function can be experimentally measured using the procedure described by Lee and coworkers [14]. These three stages are illustrated in Figure 1.

4. This process is repeated until the cell population has completely filled the scaffold or until the cells cannot migrate and divide any further [15].

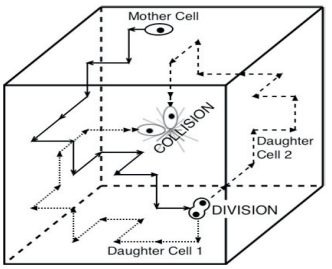

Figure 1: Illustration of persistent random movement of cells during their division cycle and prior to their collision.

When cells are seeded in a three-dimansional (3D) scaffold, they migrate in all directions, interact with each other and proliferate until they completely fill the space available to them. This assumes that enough nutrients are always available to sustain cell growth everywhere in the interior of the scaffold. To model this highly dynamic process, we consider cellular automata consisting of three-dimensional grids with $N_x \times N_y \times N_z$ total cubic computational sites [16]. Each site is a finite automaton that can exist in one of a finite number of states at each time interval. That is, a site may be either:

- empty and available for a cell to move in, or
- occupied by a cell, which is at a given point in its mitotic cycle and moves in a certain direction. No other cell can move or divide into an already occupied site.

To simulate these dynamics of tissue growth, the state of each cellular automaton takes values from a set of integer numbers that code all the required information about cell migration speed, the direction of movement, and the time remaining until the next direction change and the next cell division. Hence, every automaton has its state evolving at discrete time steps through interactions with neighboring automata. Let us consider the j^{th} automaton that contains a cell at time t^r. Its state $x_j(r)$ is specified by the following three numbers:

1. A migration index m_j: If $m_j = 1,2,\ldots,6$, then the cell is migrating in one of the six directions (east, north, west, south, up and down). If $m_j = 0$, the cell is stationary.

2. A division counter $k_{d,j}$: The time that must elapse before the cell divides is equal to $t_d = k_{d,j} * \Delta t$. For each iteration, this counter is decremented by one and the cell divides when $k_{d,j} = 0$.

3. A persistence counter $k_{p,j}$: The time that must elapse before the cell changes its direction of movement is equal to $t_p = k_{p,j} * \Delta t$. For each iteration, this counter is decremented by one and the cell turns when $k_{d,j} = 0$.

In our simulation model, random numbers are used to make compuationally coarse-grained decisions in the following functions:

1. Initial seeding of the cell population (both in terms of cell placement and cell state assignment).
2. Selection of an occupied computational site for processing.
3. Changing the direction of cell movement.
4. Placement of the new daughter cell after cell division.
5. Determining new state information for both new cells after division.
6. Determining new direction of cell movement after cell collision.

In this regard, we are interested in a sequence of independent random numbers with a specified distribution. In broad terms, this means that each number was obtained merely by chance, having nothing to do with other numbers of the sequence, and that each number has a specified probability of falling in any given range of values. Here, we assume such distribution to be *uniform*, where each possible number is equally probable [17]. This is because we have assumed in the model that the simulation will be undertaken initially without considering chemotaxis so that cell migration and division remain *unbiased* since enough nutrients are always available in the cellular space.

3 Some Theoretical Considerations

Many random number generators in use today are not very good. Users tend to avoid learning about such methods. Moreover, some of them get passed around from one programmer to the next without taking the time to understand their limitations. A very large sequence of *statistically independent* random numbers, uniformly distributed between 0 and 1 is desired. A satisfactory algorithm for random number generation was proposed by D. H. Lehmer over 60 years ago [18]. This is known as the *multiplicative linear congruential* (MLCG) algorithm [19]. The algorithm is known for its simplicity and effectiveness. It involves the careful selection of two fixed integer parameters:

1. A modulus: m, a large prime integer,
2. A multiplier: a, an integer in the range $2,3,\ldots,m-1$,

and the subsequent generation of the integer sequence $x_1,x_2,x_3\cdots$ via the iterative equation,

$$x_{n+1} = f(x_n), \text{ for } n = 1,2,\ldots,$$

where the genrating function $f()$ is defined for all x in $1,2,\ldots,m-1$ as

$$f(x) = ax \bmod m.$$

The sequence of xs must be initialized by choosing an initial integer value x_1, called the *seed*, from $1,2,\ldots,m-1$. Then, the sequence is normalized to the unit interval via division by the modulus to produce the real sequence $u_1,u_2,u_3,\ldots$, where

$$u_n = x_n/m, \text{ for } n = 1,2,\ldots.$$

It is important to note that, because m is prime, $f(x) \neq 0$ for all x in $1,2,\ldots,m-1$ and any a in $2,3,\ldots,m-1$. This is crucial because it prevents the sequence from collapsing to zero. Moreover, one

can show that the smallest and largest possible values of u are $\frac{1}{m}$ and $1-\frac{1}{m}$, respectively. Hence, the values $u = 0.0$ and $u = 1.0$ are impossible. Also, the normalization step does not affect the fundamental issue of whether or not the sequence of us appears to be random. Therefore, the issue of randomness can be entirely resolved by studying the integer sequence of xs. Hence, if the multiplier and prime modulus are properly chosen, the resulting sequence of xs can be statistically indistinguishable from a sequence drawn at random, albeit without replacement, from the set $1,2,\ldots,m-1$ [19]. Periodicity, randomness, and implementation are three central issues that we had to deal with when using a MLCG. Each effectively serves as a filter that limits the possible choices of multipliers or a given modulus.

4 Periodicity

For our simulation model, we used the *Mersenne* prime $m = 2^{31}-1 = 2,147,483,647$ as a modulus [18]. The multiplier $a = 62,089,911$ was also selected. It is a *primitive root* of m. By definition, if m is prime then a is a primitive element modulo m (or primitive root of m) iff $a^n \bmod m \neq 1$ for $n = 1,2,\ldots,m-2$ and $a^{m-1} \bmod m = 1$ [17]. The number of primitive roots for a prime m is $\phi(m-1)$, where $\phi(m-1)$ is equal to the number of integers not exceeding and relatively prime to $m-1$. The quantity $\phi(m-1)$ is known as the *Euler totient* function. There are $534,600,000$ primitive roots for the prime integer $2^{31}-1$ [19].

By substituting consecutive terms into one another, we can rewrite the generating function as

$$x_{n+1} = f(x_n) = a^n x_1 \bmod m, \text{ for } n = 1,2,3,\ldots. \tag{1}$$

Since the $\gcd(a,m) = \gcd(62089911, 2^{31}-1) = 1$ then, according to the well-known *Fermat's Theorem*, we have

$$a^{m-1} \bmod m = 1. \tag{2}$$

Hence, the sequence of xs is *periodic* with period $p = m-1$. This makes the MLCG a full-period generator. This is true if and only if a is a primitive root of m [19].

Some of the other studied generators are known as the *mixed* linear congruential generators. They are generalizations of the multiplicative generators with generating functions of the form $f(x) = (ax + c) \bmod m$, where the additive constant c satisfies the condition $c \bmod m \neq 0$. We chose not to use this type of generating function because of the cost of one extra addition per random number. The effect of c is to allow $x = 0$ as a possible value. As a result, the period of the sequence of random numbers has a length m rather than $m-1$. Other variants use powers of two as a modulus, also known as *composite moduli*. For example, when $m = 2^t$ and $c = 0$, with integer $t > 2$, the maximum attainable period is equal to $2^{t-2} = \frac{m}{4}$. This period length is achieved only by multipliers of the form $a = 4h \pm 1$, with h being an odd positive integer.

Although the latter linear congruential generators are usually faster (since the result of the modulus operation can be obtained by employing shift operations and thus, without performing division), it was observed that they display strong sequential correlations between an entry and a successor delayed by a moderately large power of two. Serious correlations have been verified for delays of 1024 and larger [20]. This can induce errors in Monte Carlo calculations where random numbers are used repeatedly in a fixed pattern. Moreover, the *variance* which is an important result of some Monte Carlo calculations would be incorrect when computed assuming that the results of separate replications are statistically independent. With respect to long-range correlations, prime moduli seem to be the safest [21, 22].

5 Randomness Quality

Because of the subjective nature of deciding which sequence of numbers is more random than others when done by visual examination, uncertainty will always exist about which random sequence is best. However, there are now statistical tests, both theoretical and empirical, that can support our choice of the multiplier a. Of all tests, the theory-based *spectral test* is one of the strongest [23]. The spectral test deals with the properties of the joint distribution of k consecutive elements of the sequence. Having a sequence $< u_n >$ of period $m-1$, the idea is to analyze the set of all $m-1$ points in k-dimensional space: $\{(u_n, u_{n+1}, \ldots, u_{n+k-1})\}$. For practical purposes, k is generally limited to $2 \leq k \leq 6$ while the latter sequence can be rewritten as $\left\{\frac{1}{m}(x, f(x), f(f(x)), \ldots, f^{k-1}(x)) \,|\, 0 < x < m\right\}$. For example, Figure 2 shows a typical small case in two dimensions for our selected values of a and m. Viewed microscopically or using a much smaller modulus, linear congruential multipliers fall on a finite number of *parallel hyperplanes*. The spectral test analyzes the *uniformity* of this lattice structure in k-space for the specified values of k. Researchers regard a multiplier as optimal if, for $2 \leq k \leq 6$ and each set of parallel hyperplanes, the *Euclidean distance* between adjacent hyperplanes does not exceed the minimal achievable distance by more than 25%. This criterion alone results in the reduction of the number of possible multipliers for prime $m = 2^{31} - 1$ to only 410 from slightly over 5×10^8 [19].

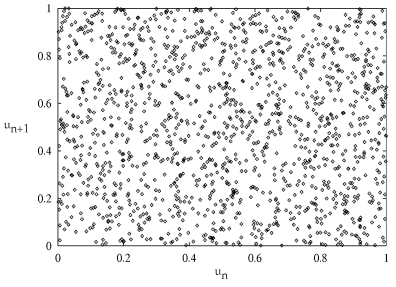

Figure 2: The two-dimensional grid formed by the first 1400 pairs of successive points (u_n, u_{n+1}) for the selected PRNG with a seed $x_1 = 123$.

To highlight the significance of the bounds imposed on the maximal distance between adjacent hyperplanes within the k-dimensional hypercube, we provide in Table 1 the computed distances for both the chosen multiplier and prime modulus, denoted by $d_k^*(a,m)$, as well as the lower bounds for $m = 2^{31} - 1$ for $2 \leq k \leq 6$, the latter as given by Fishman and Moore in [24]. For each $k \in \{2, 3, \ldots, 6\}$, the distance between adjacent hyperplanes exceeds the minimal achievable distance by only 12%, 12%, 17%, 16%, and 21% respectively, which is well within the established criterion of 25%. In particular, the authors of [24] rank this multiplier in the top five of all candidate multipliers for $m = 2^{31} - 1$. The ranking came about after performing a myriad of other tests on all the multipliers, in addition to the spectral test. As a contrasting example and using the same initial seed of 123, Figure 3 shows the three-dimensional grids formed by our generator and a similar generator with a much smaller prime modulus of $m = 61$ and two multipliers of $a = 31$ and $a = 7$, respectively. This figure illustrates the drastic change in the uniformity and spacing of the generated random numbers when carefully selected parameters are not employed..

Table 1: The Euclidean distance between adjacent hyperplanes for the selected MLCG with prime modulus $m = 2^{31} - 1$ and multiplier $a = 62,089,911$ along with its corresponding lower bound for $2 \leq k \leq 6$ [24].

k	$\mathbf{d}_{\mathbf{k}}^{*}(\mathbf{a},\mathbf{m})$	**Lower Bound** ($\forall \mathbf{a} \in \{\mathbf{2},\ldots,\mathbf{m}-\mathbf{1}\}$)
2	0.2249×10^{-4}	0.2008×10^{-4}
3	0.7733×10^{-3}	0.6905×10^{-3}
4	0.4555×10^{-2}	0.3906×10^{-2}
5	0.1280×10^{-1}	0.1105×10^{-1}
6	0.2615×10^{-1}	0.2157×10^{-1}

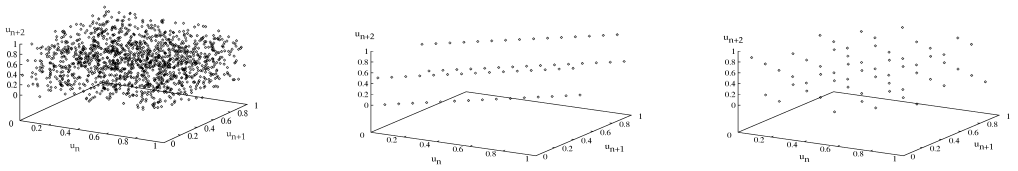

Figure 3: The three-dimensional grids formed by the first 1400 triples of successive points (u_n, u_{n+1}, u_{n+2}) for (left) the selected PRNG, (center) a PRNG with $m = 61$ and $a = 31$, and (right) a PRNG with $m = 61$ and $a = 7$, while a seed of $x_1 = 123$ is used for all three.

6 Serial Implementation

We implemented the selected MLCG using integer arithmetic in a number of high-level programming languages. Below, we give the pseudo-code for such a method:

```
Method RandMLCG(Seed1, Seed2)
Input: Seed1, an integer representing the seed of PRNG.
Outputs: RandMLCG, a real random number between 0 and 1.
         Seed2, the new seed for the next method call.
Locally declared integers: A, M, Seed1, Seed2;
A = 62089911;
M = 2147483647;
Seed2 = MOD(A * Seed1, M);
RandMLC = Seed2/Float(M);
Return
```

where *seed1* and *seed2* are defined as a global integer variables of which the former is initially chosen from the set $\{1,2,\ldots,m-2,m-1\}$. On subsequent calls to the above method, Seed1 is set equal to Seed2. Random numbers uniformly distributed between 0 and 1 can then be generated as required via repeated calls to function *RandMLCG()*. Changing the value of the seed from one run to the next results in a new sequence of random numbers that is a cyclic permutation of the other. Beacause m is represented in binary with 31 ($31 = \lceil \log_2(m) \rceil$) bits and a with 26 bits, it follows that, in the worst case, the product $a \times m$ will never exceed eight bytes. We see that $a \times m$ requires 57 bits in its 2s-complement representation. This can be amply provided in the above method, by simply declaring a, m, *seed*1,

and *seed*2 to be of length equal to 64 bits. In this case, all integers ranging from $-2^{63} \approx -10^{19}$ to $2^{63}-1 \approx 10^{19}$ are fully represented in radix two, and hence full accuracy is maintained.

When the largest positive integer that can be represented by a computer in radix two is equal to $2^{31}-1$, there is the possibility of overflow in computing $ax \bmod m$. A workaround involves decomposing the multiplier a as $a = a_1 \times a_2$ such that $a_1^2 < m$ and $a_2^2 < m$. It follows that we have $ax \bmod m = (a_1 a_2)x \bmod m = (a_1(a_2 x \bmod m)) \bmod m$. In our case, that is for $a = 62,089,911$, one possible choice is to have $a_1 = 7,813 = 13 \times 601$ and $a_2 = 7,947 = 9 \times 883$, with $a_1^2 = 61,042,969$ and $a_2^2 = 63,154,809$ with both being less than m. This is because the prime factorization of 62089911 is equal to $3^2 \times 13 \times 601 \times 883$. The condition $a_1^2 < m$ is necessary and sufficient to prevent overflow in computing $a_1 x \bmod m$ for $x \in \{1,2,\ldots,m-2,m-1\}$, as demontrated in [17].

7 Consumption of Random Numbers

For other simulations using very large data sets, the consumption of the entire period of the selected PRNG may be at risk. Nonetheless, we asked ourselves about the possibility of exhausting the period of the PRNG in our simulation experiments. Due to limitations in the size of memory available in the computing system, we were able to run our serial simulations for cellular array sizes of up to $250 \times 250 \times 250$. For each of these runs, we simulated the growth of tissue using cells moving at a maximum speed of $56 \mu m/hr$ and computed the total number of PRNs generated. We also computed the ratio R, defined as

$$R = \frac{\text{(total number of random numbers till confluence)}}{\text{(total number of sites in cellular array)}},$$

where the confluence parameter represented 99.99% of volume coverage. Our extensive simulations yielded a ratio R that never exceeded a certain number for moderate and large cellular array sizes (up to the indicated maximum above). In Table 2, we present the results of only a sample of these simulation runs showing the maximum value of R, where the corresponding size of the three-dimensional cellular array and the seeding density for which this maximum value was attained are displayed as well. Initially, cells were uniformly and randomly distributed in a cellular array with fixed boundaries. All simulations started with the same initial seed of $x_1 = 123$ for the generation of random numbers. The full set of simulation experiments are includedd and depicted in Figure 4 using seeding densities of 0.01%, 0.1%, 1%, and 10%. We observe that for three-dimensional cellular arrays having a total number of sites greater than or equal to 512×10^3 (i.e. , of dimensions larger than or equal to $80 \times 80 \times 80$), the ratio R never exceeds 40. This is an important empirical result for our simulation model. It allows us to predict that the selected MLCG will not have its full period of random numbers consumed as long as the size of the cellular array is approximately less than or equal to 53×10^6 total sites (or $375 \times 375 \times 375$).

8 Conclusion and Future Work

We overviewed and discussed in this paper our choice of pseudorandom number generation based on a prime modulus multiplicative linear congruential generator with a carefully selected multiplier. In particular, we examined some of its theoretical properties, its randomness quality and ease of implementation, in the context of our simulation model for tissue growth. The MLCG provides a good, portable, and user-controlled generator. The user has complete control on which multiplier, modulus, and initial seed to employ. In turn, these parameters afford us additional control over the periodicity, randomness, and implementation of this generator. The latter has an impact on easing the debugging of the sequential code, enhancing our confidence in the validity of the serial program and its correctness,

Table 2: Maximum values of the ratio R of random numbers generated per sequential run to the size of the 3D cellular array with fixed boundaries and a confluence parameter of 99.99%.

max(**R**)	**Array Dimensions**	**Number of Seed Cells**	**Seeding Density**
73.18	$5 \times 5 \times 5$	1	1%
51.17	$10 \times 10 \times 10$	1	0.1%
42.00	$22 \times 22 \times 22$	1	0.01%
40.03	$50 \times 50 \times 50$	12	0.01%
39.38	$80 \times 80 \times 80$	51	0.01%
39.60	$100 \times 100 \times 100$	100	0.01%
39.66	$128 \times 128 \times 128$	2097	0.01%
39.33	$160 \times 160 \times 160$	4096	0.01%
39.00	$200 \times 200 \times 200$	8000	0.01%
38.90	$250 \times 250 \times 250$	1562	0.01%

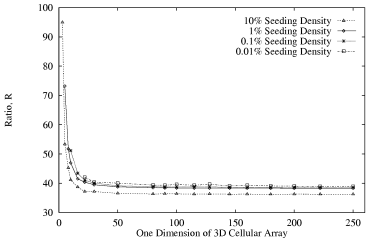

Figure 4: Variation of the ratio R of the total number of random numbers generated to the size of the 3D cellular array as one dimension of the array is changed (all three dimensions have the same value). Four plots are shown that represent initial seeding densities of 0.01%, 0.1%, 1%, and 10%, respectively.

and ensuring the reproducibility of results from one run to the next. We believe that as 64-bit computer architectures become more and more common with larger memory capacity, faster CPUs, and the need to run simulations that may consume tens to hundreds of billions of random numbers, a MLCG with a longer period will be required. The fact that the next prime Mersenne number is $2^{61} - 1$ may be considered as a possible candiddate for adoption and may present interesting design, testing, and implementation issues in our future work. One obvious benefit is that sequences with a period over 10^9 times longer than the one used here can then be obtained.

Acknowledgments

The authors would like to gratefully acknowledge the support for this work provided by the Research Centre in the College of Computer & Information Sciences (CCIS) under project number: RC1303106, as well as the Deanship of Scientific Research at King Saud University, Saudi Arabia.

References

[1] M. J. Lysaght and A. L. Hazlehurst. “Tissue Engineering: The End of the Beginning”. *Tissue Engineering*, 10(1–2):309–320, 2004.

[2] A. F. Maree and P. Hogeweg. “How Amoeboids Self-Organize Into a Fruiting Body: Multicellular Coordination in Dictyostelium Discoideum”. *Proceedings of the National Academy of Sciences, USA*, 98(7):3879–3883, 2001.

[3] G. An, Q. Mi, J. Dutta-Moscato, and Y. Vodovotz. “Agent-Based Models in Translational Systems Biology”. *Wiley Interdisciplinary Reviews: Systems Biology and Medicine*, 1(2):159–171, 2009.

[4] G. Majno and I. Joris. *Cells, Tissues and Disease: Principles of General Pathology*. Oxford University Press, Oxford, UK, 2004.

[5] M. Hwang, M. Garbey, S. A. Berceli, and R. Tran-Son-Tay. "Rule-Based Simulation of Multi-Cellular Biological Systems—A Review of Modeling Techniques". *Cellular and Molecular Bioengineering*, 2(3):285–294, 2009.

[6] K. Zygourakis, R. Bizios, and P. Markenscoff. "Proliferation of Anchorage-Dependent Contact-Inhibited Cells: I. Development of Theoretical Models Based on Cellular Automata". *Biotechnology and Bioengineering*, 38(5):459–470, August 1991.

[7] K. A. Hawboldt, N. Kalogerakis, and L. A. Behie. "A Cellular Automaton Model for Microcarrier Cultures". *Biotechnology and Bioengineering*, 43(1):90–100, January 5 1994.

[8] Y. Lee, S. Kouvroukoglou, L. V. McIntire, and K. Zygourakis. "A Cellular Automaton Model for the Proliferation of Migrating Contact-Inhibited Cells". *Biophysical Journal*, 69(10):1284–1298, October 1995.

[9] B. Ben Youssef, G. Cheng, K. Zygourakis, and P. Markenscoff. "Parallel Implementation of a Cellular Automaton Modeling the Growth of Three-Dimensional Tissues". *International Journal of High Performance Computing Applications*, 21(2):196–209, Summer 2007.

[10] G. Cheng, B. Ben Youssef, P. Markenscoff, and K. Zygourakis. "Cell Population Dynamics Modulate the Rates of Tissue Growth Processes". *Biophysical Journal*, 90(3):713–724, 2006.

[11] B. Ben Youssef and L. Tang. "Simulation of Multiple Cell Population Dynamics Using a 3-D Cellular Automata Model for Tissue Growth". *International Journal of Natural Computing Research*, 1(3):1–18, 2010.

[12] P. Frederickson, R. Hiromoto, T. L. Jordan, B. Smith, and T. Warnock. "Pseudo-Random Trees in Monte Carlo". *Parallel Computing*, 1(2):175–180, 1984.

[13] Y. Lee, P. A. Markenscoff, L. V. McIntire, and K. Zygourakis. "Characterization of Endothelial Cell Locomotion Using a Markov Chain Model". *Biochemistry and Cell Biology*, 73:461–472, 1995.

[14] Y. Lee, L. V. McIntire, and K. Zygourakis. "Analysis of Endothelial Cell Locomotion: Differential Effects of Motility and Contact Inhibition". *Biotechnology and Bioengineering*, 43(7):622–634, March 25, 1994.

[15] B. Ben Youssef. "A Visualization Tool of 3-D Time-Varying Data for the Simulation of Tissue Growth". *Multimedia Tools and Applications*, 2013. doi:10.1007/s11042-013-1657-8, in press.

[16] S. Wolfram. *Cellular Automata and Complexity: Collected Papers*. Addison-Wesley, 1994.

[17] D. E. Knuth. *The Art of Computer Programming–Volume 2: Seminumerical Algorithms*. Addison-Wesley Publishing Co., Reading, MA, second edition, 1981.

[18] D. H. Lehmer. "Mathematical Methods in Large-Scale Computing Units". In *Proceedings of a Second Symposium on Large-Scale Digital Calculating Machinery*, volume 26 of *The Annals of the Computation Laboratory of Harvard University*, pages 141–146. Harvard University Press, 1951.

[19] S. K. Park and K. W. Miller. "Random Number Generators: Good Ones are Hard to Find". *Communications of the ACM*, 31(10):1192–1201, October 1988.

[20] O. E. Percus and M. H. Kalos. "Random Number Generators for MIMD Parallel Processors". *Journal of Parallel and Distributed Computing*, 6(3):477–497, June 1989.

[21] A. De Matteis and S. Pagnutti. "Parallelization of Random Number Generators and Long-Range Correlations". *Numerische Mathematik*, 53:595–608, 1988.

[22] S. L. Anderson. "Random Number Generators on Vector Supercomputers and Other Advanced Architectures". *SIAM Review*, 32(2):221–251, June 1990.

[23] D. P. Kroese, T. Taimre, and Z. I. Botev. *Handbook of Monte Carlo Methods*. Wiley Series in Probability and Statistics. John Wiley and Sons, 2011.

[24] G. S. Fishman and L. R. Moore. "An Exhaustive Analysis of Multiplicative Congruential Random Number Generators with Modulus $2^{31} - 1$". *SIAM Journal of Scientific and Statistical Computing*, 7(1):24–45, January 1986.